OTOT-A2

Name: Erin May Gunawan

Matric. No: A0200765L

GitHub Link: A2-A3

Demo Link: A2

Task A2.1

Cluster Creation

1. Ensure `kubectl` and `kind` is installed on your machine.
2. To create a cluster, open your terminal, navigate to the root of this project, and run:

```
kind create cluster --name kind-1 --config k8s/kind/cluster-config.yaml
```

3. Wait a few minutes until the cluster is set up, once it is set up, you will see a confirmation message:

```
Creating cluster "kind-1" ...
✓ Ensuring node image (kindest/node:v1.25.0) 🖼
✓ Preparing nodes 📦 📦 📦 📦
✓ Writing configuration 📜
✓ Starting control-plane 🕹️
✓ Installing CNI 🔌
✓ Installing StorageClass 💾
✓ Joining worker nodes 🚜
Set kubectl context to "kind-kind-1"
You can now use your cluster with:

kubectl cluster-info --context kind-kind-1
```

4. To ensure that the cluster is created correctly, you can run the following commands:

```
# Inspect that the cluster nodes are running on docker
docker ps -a

# Inspect the nodes via kubectl
kubectl get nodes

# Inspect the cluster-info
kubectl cluster-info
```

When running the `kubectl cluster-info`, ensure that the control-plane is running on `localhost` or `127.0.0.1` and the port matches the `kind-control-plane`'s exposed port.

Task A2.2

Deploy Image to DockerHub

1. Sign up for an account at DockerHub. Create a public repository.
2. Tag existing image and push to DockerHub

```
# Login to docker
docker login

# Tag and push to DockerHub
docker tag <image-name> <docker-username>/<image-name>[:tag]
docker push <docker-usename>/<image-name>[:tag]
```

3. Verify that the repository exists on DockerHub by navigating to `https://hub.docker.com/repository/docker/<docker-username>`

Create Deployment

1. Create a manifest file (`.yaml`) in `k8s/manifests/` and fill the contents with the following:

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: backend
labels:
  app: backend
spec:
replicas: 3
selector:
  matchLabels:
    app: backend
template:
  metadata:
    labels:
    app: backend
  spec:
    containers:
      - name: backend
        image: <docker-username>/<docker-repo>[:tag]
        ports:
          - name: http
            containerPort: 80
        resources:
          limits:
```

```
            cpu: 40m
            memory: 100Mi
```

Do take note of the indentation as `.yaml` files rely on those for parsing information.

In this case, the `template.spec.containers[0].image` would be `erinmayg/nginx-sample` (the image from A1.2)

2. Run the following command to deploy the Deployment to the cluster:

```
kubectl apply -f <filepath to yaml>
```

Assuming you are in the root folder of this project, the command you need to run would be:

```
kubectl apply -f k8s/manifests/backend-deploy.yaml
```

Create Service

1. Create a manifest file (`.yaml`) in `k8s/manifests/` and fill the contents with the following:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: backend
    name: backend
  spec:
    selector:
      app: backend
    type: ClusterIP
    ports:
      - name: http
        port: 8080
        protocol: TCP
        targetPort: http
```

2. Run the following command to deploy the Deployment to the cluster:

```
kubectl apply -f <filepath to yaml>
```

Assuming you are in the root folder of this project, the command you need to run would be:

```
kubectl apply -f k8s/manifests/backend-service.yaml
```

3. Check if the service is running correctly by running:

```
kubectl port-forward service/backend <portNo>:8080
```

You can open `https://localhost:portNo` in your browser, it should display the nginx webpage.

Create Ingress Controller

1. Ensure all nodes are ingress-enabled

```
kubectl get nodes -L ingress-ready
```

It should output all the nodes in the cluster (same output as `kubectl get nodes`).

2. Deploy the `nginx-ingress-controller`

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-
nginx/main/deploy/static/provider/kind/deploy.yaml

# Check if deployed
kubectl -n ingress-nginx get deploy -w
```

The `-w` flag stands for `--watch`, once the deployment is `READY` (i.e. `1/1`) you can just stop the command.

Create Ingress

1. Create a manifest file for the Ingress object in `k8s/manifests/`

```
# backend-ingress.yaml

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: backend
  labels:
    app: backend
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
    - http:
        paths:
          - path: /app
```

```
            pathType: Prefix
            backend:
              service:
                name: backend
                port:
                  name: http
```

2. Deploy the Ingress to the cluster

```
kubectl apply -f k8s/manifests/backend-ingress.yaml
```

3. Check if Ingress is deployed

```
kubectl get ingress
```

It should output

```
NAME      CLASS    HOSTS   ADDRESS     PORTS   AGE
backend   <none>   *       localhost   80      112m
```

4. Check if Ingress is running correctly by opening https://localhost:80/app in your browser. It should display the nginx webpage.